**Building Synthetic Voices**
Limited domain synthesis
Computer Speech Processing Group Project PartA
Due : In next class

---

# Telling the time



Festival includes a very simple little script that speaks the current time (@file{festival/examples/saytime}). This section explains how to replace the synthesizer used from this script with one that talks with your own voice. This is an extreme example of a limited domain synthesizer but it is a good example as it allows us to give a walkthrough of the stages involved in building a limited domain synthesizer. This example is also small enough that it can be done in well under an hour.

Following through this example will give a reasonable understanding of the relative importance of many important steps in the voice building process.

The following tasks are required:

- Designing the prompts
- Customized the synthesizer front end
- Recording the prompts
- Autolabeling the prompts
- Building utterance structures for recorded utterances
- Extracting pitchmark and building LPC coefficients
- Building a clunit based synthesizer from the utterances
- Testing and tuning

Before starting set the environment variables `FESTVOXDIR` and `ESTDIR` to the directories which contain the festvox distribution and the Edinburgh Speech Tools respectively. Under bash and other good shells this may be done by commands like

export FESTVOXDIR=/home/awb/projects/festvox
export ESTDIR=/home/awb/projects/1.4.3/speech_tools

In earlier releases only offered a command line based method for building voices and limited domain synthesizers. In order to make the process easier and less prone to error we have introduced and graphical front end to these scripts. This front end is called `pointyclicky` (as it offers a pointy-clicky interface). It is particularly useful in the actual prompting and recording. Although `pointyclicky` is the recommend route in the section we go through the process step by step to give a better understanding of what is required and where problems may lie that require attention.

A simple script is provided setting up the basic directory structure and copying in some default parameter files. The festvox distribution includes all the setup for the time domain. When building for your domain, you will need to provide the file `etc/DOMAIN.data` contains your prompts (as described below).

```
mkdir ~/data/time
cd ~/data/time
$FESTVOXDIR/src/ldom/setup_ldom cmu time awb
```

As in the definition of diphone databases we require three identifiers for the voice. These are (loosely) institution, domain and speaker. Use `net` if you feel there isn't an appropriate institution for you, though we have also use the project name that the voice is being build for here. The domain name seems well defined. For speaker name we have also used style as opposed to speaker name. The primary reason for these to so that people do not all build limited domain synthesizer with the same thus making it not possible to load them into the same instance of festival.

This setup script makes the directories and copies basic scheme files into the `festvox/` directory. You may need to edit these files later.

# Designing the prompts

In this `saytime` example the basic format of the utterance is

The time is now, EXACTNESS MINUTE INFO, in the DAYPART.

For example

The time is now, a little after five to ten, in the morning.

In all there are 1152 (4x12x12x2) utterances (although there are three possible day info parts (morning, afternoon and evening) they only get 12 hours, 6 hours and 6 hours respectively). Although it would technically be possible to record all of these we wish to reduce the amount of recording to a minimum. Thus what we actually do is ensure there is at least one example of each value in each slot.

Here is a list of 24 utterances that should cover the main variations.

The time is now, exactly five past one, in the morning
The time is now, just after ten past two, in the morning
The time is now, a little after quarter past three, in the morning
The time is now, almost twenty past four, in the morning
The time is now, exactly twenty-five past five, in the morning
The time is now, just after half past six, in the morning
The time is now, a little after twenty-five to seven, in the morning
The time is now, almost twenty to eight, in the morning

The time is now, exactly quarter to nine, in the morning
The time is now, just after ten to ten, in the morning
The time is now, a little after five to eleven, in the morning
The time is now, almost twelve.
The time is now, just after five to one, in the afternoon
The time is now, a little after ten to two, in the afternoon
The time is now, exactly quarter to three, in the afternoon
The time is now, almost twenty to four, in the afternoon
The time is now, just after twenty-five to five, in the afternoon
The time is now, a little after half past six, in the evening
The time is now, exactly twenty-five past seven, in the evening
The time is now, almost twenty past eight, in the evening
The time is now, just after quarter past nine, in the evening
The time is now, almost ten past ten, in the evening
The time is now, exactly five past eleven, in the evening
The time is now, a little after quarter to midnight.

These examples are first put in the prompt file with an utterance number and the prompt in double quotes like this.

(time0001 "The time is now ...")
(time0002 "The time is now ...")
(time0003 "The time is now ...")
...

These prompt should be put into `etc/DOMAIN.data`. This file is used by many of the following sub-processes.

# Recording the prompts

Record a few examples on the PC to see how much noise is being picked up by the mike. For example try the following

$ESTDIR/bin/na_record -f 16000 -time 5 -o test.wav -otype riff

This will record 5 seconds from the microphone in the machine you run the command on. You should also do this to test that the microphone is plugged in (and switched on). Play back the recorded wave with `na_play` and perhaps play with the mixer levels until you get the least background noise with the strongest spoken signal. Now you should display the waveform to see (as well as hear) how much noise is there.

$FESTVOXDIR/src/general/display_sg test.wav

This will display the waveform and its spectrogram. Noise will show up in the silence (and other) parts.

There a few ways to reduce noise. Ensure the microphone cable isn't wrapped around other cables (especially power cables). Turning the computer 90 degrees may help and repositioning things can help too. Moving the sound board to some other slot in the machine can also help as well as getting a different microphone (even the same make).

There is a large advantage in recording straight to disk as it allows the recording to go directly into right files. Doing off-line recording (onto DAT) is better in reducing noise but transferring it to disk and segmenting it is a long and tedious process.

Once you have checked your recording environment you can proceed with the build process.

First generate the prompts with the command

festival -b festvox/build_ldom.scm '(build_prompts "etc/time.data")'

and prompt and record them with the command

bin/prompt_them etc/time.data

You may or may not find listening to the prompts before speaking useful. Simply displaying them may be adequate for you (if so comment out the `na_play` line in `bin/prompt_them}`.

# Autolabeling the prompts

The recorded prompt can be labeled by aligning them against the synthesize prompts. This is done by the command

bin/make_labs prompt-wav/*.wav

If the utterances are long (> 10 seconds of speech) you may require lots of swap space to do this stage (this could be fixed).

Once labeled you should check that they are labeled reasonable. The labeler typically gets it pretty much correct, or very wrong, so a quick check can often save time later. You can check the database using the command

emulabel etc/emu_lab

Once you are happy with the labeling you can construct the whole utterance structure for the spoken utterances. This is done by combining the basic structure from the synthesized prompts and the actual times from the automatically labeled ones. This can be done with the command

festival -b festvox/build_ldom.scm '(build_utts "etc/time.data")'

# Extracting pitchmarks and building LPC coefficients

Getting good pitchmarks is important to the quality of the synthesis, see the Section called *Extracting pitchmarks from waveforms* in the Chapter called *Basic Requirements* for more detailed discussion on extrating pitchmarks from waveforms. For the limited domain synthesizers the pitch extract is a little less crucial that for diphone collection. Though spending a little time on this does help.

If you have recorded EGG signals the you can use `bin/make_pm` from the `.lar` files. Note that you may need to add (or remove) the option `-inv` depending on the updownness of your EGG signal. However so far only the CSTR larygnograph seems to produce inverted signals so the default should be adequate. Also note the parameters that specify the pitch period range, `-min` and `max` the default setting are suitable for a male speaker, for a female you should modify these to something like

-min 0.0033 -max 0.0875 -def 0.005

The changing from a range of (male) 200Hz-80Hz with a default of 100Hz, to a female range of 300Hz-120Hz and default of 200Hz.

If you don't have an EGG signal you must extract the pitch from the waveform itself. This works though may require a little modification of parameters, and it is computationally more expensive (and wont be as exact as from an EGG signal). There are two methods, one using Entropic's `epoch` program which work pretty well without tuning parameters. The second is to use the free Speech Tools program `pitchmark`. To use `epoch` use the program

bin/make_pm_epoch wav/*.wav

To use `pitchmark` use the command

bin/make_pm_wave wav/*.wav

As with the EGG extraction `pitchmark` uses parameters to specify the range of the pitch periods, you should modify the parameters to best match your speakers range. The other filter parameters also can make a different to the success. Rather than try to explain what changing the figures mean (I admit I don't fully know), the best solution is to explain what you need to obtain as a result.

Irrespective of how you extract the pitchmarks we have found that a post-processing stage that moves the pitchmarks to the nearest peak is worthwhile. You can achieve this by

bin/make_pm_fix pm/*.pm

At this point you may find that your waveform file is upside down. Normally this wouldn't matter but due to the basic signal processing techniques we used to find the pitch periods upside down signals confuse things. People tell me that it shouldn't happen but some recording devices return an inverted signal. From the cases we've seen the same device always returns the same

form so if one of your recordings is upside down all of them probably are (though there are some published speech databases e.g. BU Radio data, where a random half are upside down).

In general the higher peaks should be positive rather than negative. If not you can invert the signals with the command

```
for i in wav/*.wav
do
  ch_wave -scale -1.0 $i -o $i
done
```

If they are upside, invert them and re-run the pitch marking. (If you do invert them it is not necessary to re-run the segment labeling.)

Power normalization can help too. This can be done globally by the function

```
bin/simple_powernormalize wav/*.wav
```

This should be sufficient for full sentence examples. In the diphone collection we take greater care in power normalization but that vowel based technique will be too confused by the longer more varied examples.

Once you have pitchmarks, next you need to generate the pitch synchronous MELCEP parameterization of the speech used in building the cluster synthesizer.

```
bin/make_mcep wav/*.wav
```

# Building a clunit based synthesizer from the utterances

Building a full clunit synthesizer is probably a little bit of over kill but the technique basically works. See the Chapter called *Unit selection databases* for a more detailed discussion of unit selection technique. The basic parameter file `festvox/time_build.scm` is reasonable as a start.

```
festival -b festvox/build_ldom.scm '(build_clunits "etc/time.data")'
```

If all goes well this should create a file `festival/clunits/cmu_time_awb.catalogue` and set of index trees in `festival/trees/cmu_time_awb_time.tree`.

# Testing and tuning

To test the new voice start Festival as

```
festival festvox/cmu_time_awb_ldom.scm '(voice_cmu_time_awb_ldom)'
```

The function `(saytime)` can now be called and it should say the current time, or `(saythistime "11:23")`.

Note this synthesizer can *only* say the phrases that it has phones for which basically means it can only say the time in the format given at the start of this chapter. Thus although you can use `SayText` it will only synthesis words that are in the domain. That's what limited domain synthesis is.

A full directory structure of this example with the recordings and parameters files is available at http://festvox.org/examples/cmu_time_awb_ldom/. And an on-line demo of this voice in that directory is available at http://festvox.org/examples/cmu_time_awb_ldom/.



**Some useful pointers for guidance:**

http://www.festvox.org/bsv/book1.html